PRIMEBASE
SYSTEM OVERVIEW

2nd Edition
July 2008
PrimeBase Systems GmbH
Max-Brauer-Allee 50 - D – 22765 Hamburg - Germany
www.primebase.com - e-mail: info@primebase.com

PRIMEBASE SYSTEM OVERVIEW

PrimeBase is an integrated, open system for the creation of internet, intranet and wireless applications (Web applications). These applications are programs based on Internet standards, and often have Web-browser hosted user interfaces. PrimeBase is integrated, because it includes all the components you need to build such an application. PrimeBase is open, because each component, together with the PrimeBase Virtual Machine, forms a unit that can be used on its own and in conjunction with 3rd party components. For this purpose, PrimeBase implements numerous interfaces and other standards including Java, SQL, HTTP, XML, ODBC, JDBC, DAM, EOF and many more. The main components of the PrimeBase System are illustrated in Figure 1 below.

1. Introduction

This document describes the PrimeBase System as a whole. It serves as pre-requisite to all other documentation on PrimeBase. The various PrimeBase components work together to provide a complete solution. In addition, all components are used in conjunction with the PrimeBase Virtual Machine (see the section "Design of the PBVM" for details). It is therefore important that the user, system administrator or programmer understands the function of each PrimeBase component. Knowledge of the PrimeBase System will help you to assess the possibilities PrimeBase offers you to meet your IT needs.

As it will become clear, these possibilities go way beyond the creation of Web applications. PrimeBase can act as middleware, tying diverse systems together. PrimeBase can also act as an e-Business platform or as the basis of an industrial controller system.

In addition to this we offer numerous solutions based on the PrimeBase technology, including PrimeBase Job Market, PrimeBase Auction and PrimeBase E-Commerce. Further solutions are offered together with numerous partners. PrimeBase solutions can be used "as is", or serve as the foundation for your own further development and value added re-sale. Because of the integrated/open nature of PrimeBase these solutions together with extensions can also be shipped as a turn-key system.

Besides performance, stability and reliability that may be expected from a system such as PrimeBase, the watchword of PrimeBase is "simplicity". Simplicity today is often taken to mean low-functionality. However, the systems of tomorrow, as is the case with PrimeBase, will have to offer both: simplicity and high-functionality.

The simplicity of the PrimeBase System means the following for you:

- Easy, fast and cost effective, "copy-and-start" installation.
- Low maintenance costs. All PrimeBase components are designed to run without constant monitoring.
- Minimal training effort. Since PrimeBase is easy to understand users are sooner productive.
- Moderate hardware requirements in comparison to other systems.

Today we are faced with numerous choices in the world of computing technology. Many systems and standards are offered which perform the same task. What should then your criteria of choice be? We believe, if you have a choice, then choose "the simplest" (the system you are building is probably complex enough).

In the same way PrimeBase strives to choose the standards that are integrated into the system.

2. System Components

Figure 1 illustrates the various components of the PrimeBase System and the data low between the components.

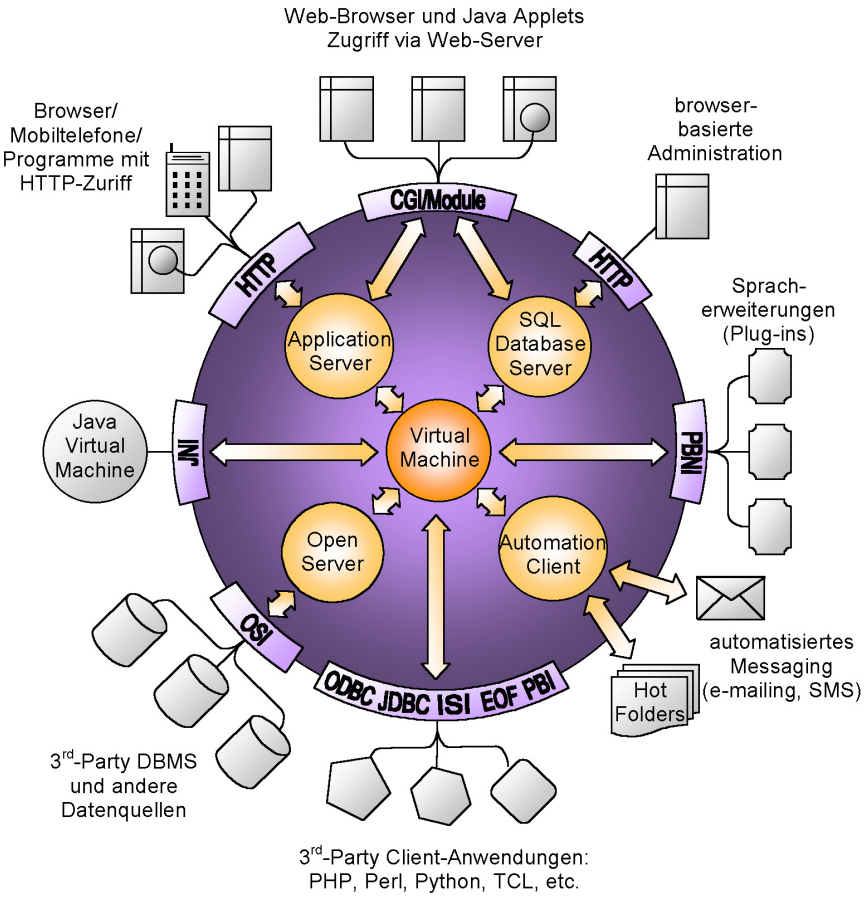

Figure 1: PrimeBase – an integrated, open system.

The diverse interfaces that integrate PrimeBase into existing and 3rd party systems are pictured around the edge of the PrimeBase System.

As illustrated, the hub of the PrimeBase System is the PrimeBase Virtual Machine. Positioned around the hub are the four other main components that make up the PrimeBase System: the PrimeBase SQL Database Server, the PrimeBase Application Server, the PrimeBase Open Server and the PrimeBase Automation Client.

The PrimeBase Virtual Machine

The PrimeBase Virtual Machine (PBVM) is the central processing unit of the PrimeBase System, and is therefore of primary importance to the entire system. The PBVM is responsible for transporting and manipulating all forms of data. It is also used to bind external execution units into the PrimeBase System. This includes seamless integration with the Java Virtual Machine (using the Java Native Interface), and the inclusion of native language extensions that add further functionality in the form of "plug-ins" (using the PrimeBase Native Interface).

The PrimeBase Virtual Machine is programmable. It supports all common programming constructs and statements and the creation of classes and objects in a manner similar to Java. The programming language, known as PrimeBaseTalk (PBT) also includes the full SQL (Structured Query Language) standard for database access. All database data types are native to PBT, including types such as NUMERIC, UNICODE, BLOBs and database NULL values. This makes the PBVM ideal for the exchange, conversion and filtering of business and media data.

Because of the central importance of the PBVM to the PrimeBase System, the functioning of this component is described in more detail in the section "Design of the PBVM".

The PrimeBase SQL Database Server

The PrimeBase SQL Database Server (PBDS) is a standard relational database system. Besides support for the ANSI SQL '92 standard PBDS includes excellent support for Unicode, Binary and Character Large Objects and a full-text indexing system. These features make PBDS ideal for development of dynamic Internet and Intranet Web sites.

In addition to this, the PrimeBase SQL Database Server supports Web-based administration as illustrated in Figure 1. Databases can be created and modified, and many other administration tasks performed from an easy-to-use browser interface. This interface is called the PBDS Admin Server.

Based on classic client/server technology, PBDS can be used as a stand-alone DBMS. Access to the PBDS provided by the PrimeBase Virtual Machine that supports all common database access standards, including:

- ODBC – Microsoft's Open Database Connectivity,
- JDBC – Sun Microsystems' Java Database Connectivity,
- ISI – PrimeBase Item Stream Interface, based on Apple's Data Access Manager (DAM/DAL) standard.
- EOF Adaptor – Apple's Enterprise Object Framework Adaptor and,
- PBI – the PrimeBase Interface. An interface that includes all the functionality of ODBC, without the complexity.

In this way, the PrimeBase Virtual Machine acts as (what is commonly known as) the client software for the PrimeBase SQL Database Server. As we have seen, operating as a database client software is only one aspect of the PrimeBase Virtual Machine, but it is an important aspect.

This means that, for example, when you install the PrimeBase ODBC Driver (in order to run traditional client/server applications, such as MS Access or Delphi) the PrimeBase Virtual Machine will be installed on your machine as well. The ODBC Driver passes the data to the PBVM that compiles the SQL statements and sends them to the PrimeBase SQL Database Server or Open Server.

The PrimeBase Application Server

A further key component of the PrimeBase System is the PrimeBase Application Server (PBAS). An application server is so called, because it runs applications on behalf of a client program. In the case of the PrimeBase Application Server this is a Web-browser or any other internet/intranet "user agent", such as a Java applet, or even some other application server.

Application serving adds a 3rd layer to the traditional client/server-computing model. For this reason application servers provide, what is known as, a 3-tier architecture. The three tiers are the user interface (client program), the application (business logic) and the database (persistent storage).

This architecture has become very popular, particularly with the advent of the Internet, because it has many advantages. In particular, it centralizes administration, maintenance, access control and security of an application.

The PrimeBase Application Server supports the Internet transport standard HTTP and also works together with Web servers to forge the link between Internet user-agents and the PrimeBase System. In doing this, it uses a technology unique to PrimeBase known as "LiveTags". LiveTags™ is a technique by which standard HTML or XML tags are filled or replaced with dynamic content. Therefore code and layout of any application are being separated totally. This is a significant ease to administrating applications.

Through support for the Common Gateway Interface (CGI) and support for native modules and shared libraries, PBAS runs with numerous Web servers, including

Apache, MS Internet Information Server, iPlanet (formerly Netscape) Web server and WebSTAR.

The PrimeBase Application Server relies on the PrimeBase Virtual Machine, to execute the applications it hosts. These applications, in turn have complete access to the entire PrimeBase system and 3rd party systems as provided by the virtual machine.

For further details on the functioning of the PrimeBase Application Server, please refer to the PrimeBase Application Server User's Guide.

PrimeBase Replication Server

The PrimeBase Replication Server is used to maintain and constantly synchronize additional active or passive instances of databases on different PrimeBase SQL Database Servers.

PrimeBase Replication Server is available for MacOS, Mac OS X, Linux, AIX, Solaris, and Windows.

Because with replication several database servers can host the same database simultaneously, client applications can access any of the servers, thus spreading the load over all database server instances.

With more than one database server hosting the database, if one server fails, the client application can switch over to one of the remaining database servers to keep on working correctly.

With servers being geographically distributed, local database servers can increase database access performance (by avoiding possibly slow long distance internet connections) as well as reduce communication costs, because the local database servers will be used.

PrimeBase Replication Server ships with a standard replication mechanism and can be adapted to suit custom replication requirements easily.

Bi-directional replication is fully supported, so that the database replicas can not only be used for reading but also for updating the contents of the databases. The changes done to any replica will then be forwarded to any other replica, if configured respectively.

PrimeBase Media Cache Manager

The PrimeBase Media Cache Manager (PBMCM) has been developed for use with high load PrimeBase Web applications. It drastically reduces load on the PrimeBase SQL Database Servers.

Very often a Web application stores images and binary data in a database using BLOBs (binary large objects). Upon request the Web application retrieves these BLOBs from the database and delivers them to the client.

If the Web application is successful and under high load, retrieving these BLOBs can put a large strain on the database server. The reason is that it has to load the BLOB data from the filesystem and send it to the requesting Web application for every individual request. In this situation, given enough requests, even the fastest database server can become a bottleneck.

The idea behind the PrimeBase Media Cache Manager is to completely relieve the database server of the duty to retrieve and deliver the BLOBs requested by Web clients, as well as to cache the retrieved BLOBs as close to the client as possible (usually on the Web / Application Server computer).

PrimeBase Media Cache Manager consists of two components:

- PrimeBase Media Cache Manager (Apache module)
- Media Cache Server

Consider the following diagram (figure 2 on page 5) which outlines a possible configuration of Web server and the PrimeBase products Application Server, SQL Database Server and Media Cache Manager:

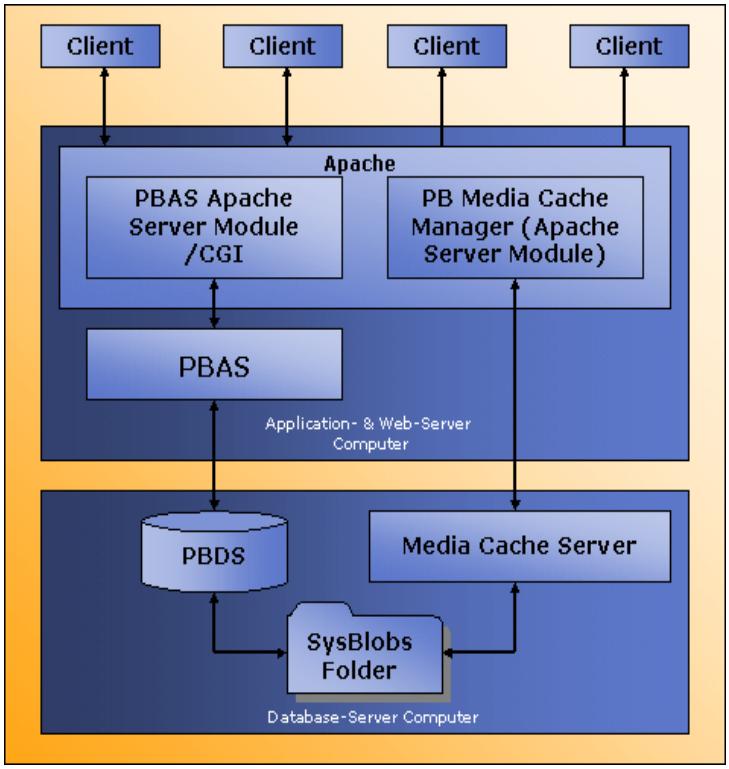

Figure 2: A typical configuration for a web application using the PrimeBase Media Cache Manager

Note that the PrimeBase Media Cache Manager is an Apache module and that the Media Cache Server runs as a separate process on the database server computer.

In general a request to the Web application hosted in PBAS will be handled in three steps:

- A Web browser sends a request for a page to the Web server.
- The request gets forwarded via the PBAS Apache module to the PrimeBase Application Server.
- PBAS (typically, but not necessarily exclusively) generates an HTML reply page which gets transferred back to the Web browser via the PBAS Apache module.

The HTML reply page generation is of particular interest at this point, because URLs referencing images, spreadsheets, and other BLOB data are dynamically placed inside the reply page by PBAS. To understand why this generation of URLs is of interest, we first need to look at how PBAS does URL generation by default.

How does PBAS handle BLOB references inserted (“printed”) into the HTML page?

When the Web application selects a BLOB from a database (via SQL SELECT database statement) and then "prints" it into an HTML page, PBAS automatically generates a file containing the raw data of the BLOB inside the "temp" folder inside its "docs" directory.

Additionally PBAS places a reference to this temporary file inside the generated HTML page. The form of this reference is configurable and defaults to
<img src='*.gif'>[1]

This way of generating BLOB references in PBAS needs to be changed to let

1. The '*' character gets replaced by the path and name of the temporary file.

Apache recognize and then forward these BLOB requests to the Media Cache Manager, which is a precondition for the following steps.

When using the PrimeBase Media Cache Manager, PBAS is configured to insert URLs which contain unique BLOB identifiers. Also the generated URLs contain information that allow Apache to recognize the requests for these URLs to forward them to the Media Cache Manager.

Whenever PBMCM receives a request forwarded to it by Apache, it checks wether the BLOB with the requested identifier is located in its local cache folder. If it does not exist there, Media Cache Manager requests the BLOB from the Media Cache Server running on the database server computer. Once in the local cache folder, the BLOB gets returned to the Web browser by PBMCM.

To further reduce load on the database server, aside from caching the BLOBS on the Web / Application Server computer, the Media Cache Server on the database server computer directly reads the BLOBs from the filesystem without ever contacting the database server.

In this way the PrimeBase SQL Database Server is completely relieved of the task to read and send BLOB data.

PrimeBase Media Cache Manager is available for Mac OS X, Linux, Solaris, AIX and Windows.

The PrimeBase Open Server

The PrimeBase Open Server (PBOS) provides access to 3rd party database management systems (DBMSs) and other data sources. This is done by creating a PrimeBase Open Server Interface (PBOSI) compliant plug-in for the data source.

The plug-in is responsible for the translation of data and SQL statements into a DBMS specific form as required by the 3rd party system. In this way, programs can be written for PrimeBase, which are DBMS independent.

The PrimeBase Open Server can run as a stand-alone application, but it is also part of the PrimeBase Virtual Machine. This means that the PBVM and the stand-alone Open Server application can load a PBOSI Plug-in.

Which configuration you choose will depend on your system. Direct access via the PBVM is recommended for maximum efficiency, however this will not be possible, if the PBOSI Plug-in for the target DBMS is not available for the platform that the PBVM will be running on.

In this case, you can make use of the cross-platform capabilities of the PrimeBase System. Do this by setting up a PBOS application on the same platform as the target DBMS (or any other platform supported by the DBMS), and use the PBVM to transport that data over the network to the DBMS machine.

During development a stand-alone PBOS application is also very useful. In development mode the PBOS application can trace all activity that occurs between the PBVM and the 3rd Party DBMS. This helps developers to find incorrect statements and other problems quickly.

The PrimeBase Automation Client

The PrimeBase Automation Client (PBAC) is the "worker" of the PrimeBase System. PBAC is responsible for performing periodic tasks and batch jobs on behalf of the rest of the system.

As illustrated in Figure 1, this includes such tasks as sending e-mails and, so-called, "hot-foldering". A hot-folder is a directory in the file system that is constantly monitored by the automation client. When data or code arrives in the folder, it is immediately processed or loaded and executed by the automation client.

It is also possible to set a timer in the automation client to perform a task at regular intervals. This may include, for example, a regular backup of the database.

In the same way as the PrimeBase Application Server, the PrimeBase Automation Client uses the PBVM to execute the task at hand. In other words, PBAC determines when a task should be started and the data or code that is to be processed

but lets the PBVM do the "number crunching".

In this sense, PBAC acts as a "shell" for the PrimeBase Virtual Machine. And, just like other shells (notable UNIX command shells), PBAC also has an interactive mode which allows you to enter code-fragments to be executed by the PBVM directly at the console.

This includes SQL and all other PBT statements supported by the PBVM, including the creation and usage of PBT and Java objects. Therefore, PBAC can be used by programmers to test SQL statements and other program components during the development of an application.

Data from any data source connected to the PrimeBase System can be retrieved and displayed in this interactive mode. This is possible because, through the PBVM, the PrimeBase Automation Client also has complete access to all other components of the PrimeBase System.

3. Design of the PrimeBase Virtual Machine

The PrimeBase Virtual Machine, as the central component of the PrimeBase System is the key to understanding how the system works. As mentioned before, all other components of the PrimeBase System require the PBVM in some form or another. In this section we explain the relationship of PBVM .

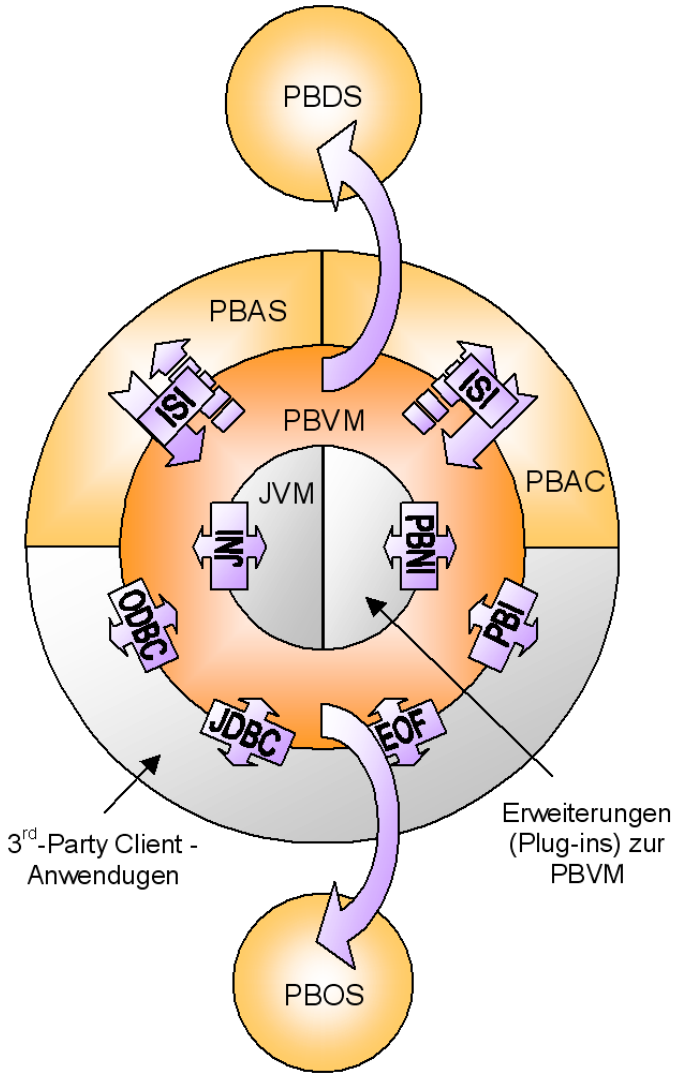

Figure 3: Data Flow to and from the PBVM

to the other components, and we highlight some of the details of its design.

Flow of Data

As we zoom in on the PBVM, the first thing we notice is that the data flowing to and from the other components of the system are of a different nature depending on the type of component. This is clearly illustrated in figure 3 on page 7. The data flow between the components can be classified into three groups:

- Shared Library Call Interfaces: The ODBC, JDBC, EOF Adaptor, PBI (PrimeBase Interface), JNI (Java Native Interface) and the PBNI (PrimeBase Native Interface) interfaces are standard shared library call level interfaces. Data is exchanged directly through these calls. These interfaces are used by various 3rd party client applications.
- Item Stream Interfaces: This interface is used by the PrimeBase Application Server and the PrimeBase Automation Client. The "item stream" is the standard output mechanism of the PBVM. Each item in the stream has a type, length, scale (for fixed-point values), flags and data. The PrimeBase Item Stream Interface may also be used by 3rd party client applications (not illustrated in Figure 3).
- Network Transport Interfaces: As illustrated in Figure 3 the PBVM exchanges data with the PrimeBase SQL Database Server and the PrimeBase Open Server using standard network protocols. This includes TCP/IP, AppleTalk and shared memory communications if the virtual machine and the server are on the same machine. Requests are sent to the server in the form of SQL statements, and the virtual machine receives pages of data as the reply from the server.

Note, in regard to the network transport interfaces: the PBVM and the PrimeBase SQL Database Server can be linked together to create a "runtime" version of PBAS. In this case, the interface is actually a call-level interface and is known as the Internal/Runtime communications protocol. In addition to this, the PBVM is always linked with a runtime version of PBOS. The Internal/Runtime communications protocol can therefore, also be used to access the PrimeBase Open Server Interface.

Component Hosting

Another aspect of the PrimeBase Virtual Machine illustrated by Figure 3 is the concept of "component hosting". In particular, the PBVM is "hosted" by the application that accesses it using the call-level interfaces, ODBC, JDBC, EOF, PBI and ISI. The PBVM in turn hosts the Java Virtual Machine and other native extensions that are accessed through the JNI and PBNI interfaces.

The hosting components in Figure 3 are illustrated as surrounding the hosted components. In practice, the hosted component is always a dynamic linked library (also known as a shared object or code fragment depending on the computer platform). The outer-most host component must always be an application. This is usually the PrimeBase Application Server, the PrimeBase Automation Client or some other 3rd party client application as pictured in Figure 3.

The hosted component forms a, so-called, "runtime environment" which has its own data area and runs independently of the hosting component. Synchronization of the host and the hosted components occurs over the common interface.

Sessions

An important aspect of the design of the PrimeBase Virtual Machine is the session concept. This becomes clear as we take a closer look at the operation of the virtual machine itself. Figure 4, illustrates the internal design of the PBVM.

A session has its own memory address space within the runtime environment of the virtual machine. Each session has a thread of execution, which originates in the hosting application. A session can also be referred to as an application or "program instance".

The virtual machine runs multiple instances of the same PBT program, each in its own session. The session boundaries ensure that each instance of the program runs independently from the others. Program instances cannot affect each other, because each session has its own address space. This ensures that each program

instance is protected from the failure of other program instances.

Creation and Termination of Sessions

The PBVM provides API level calls for the creation and termination of sessions. When a session is created, it loads the code required to run the program. Loaded code is compiled, optimized and stored in a platform independent "byte-code", ready for execution. This is called the "boot phase" of the session. Further code can be submitted for execution or storage using additional API level calls.

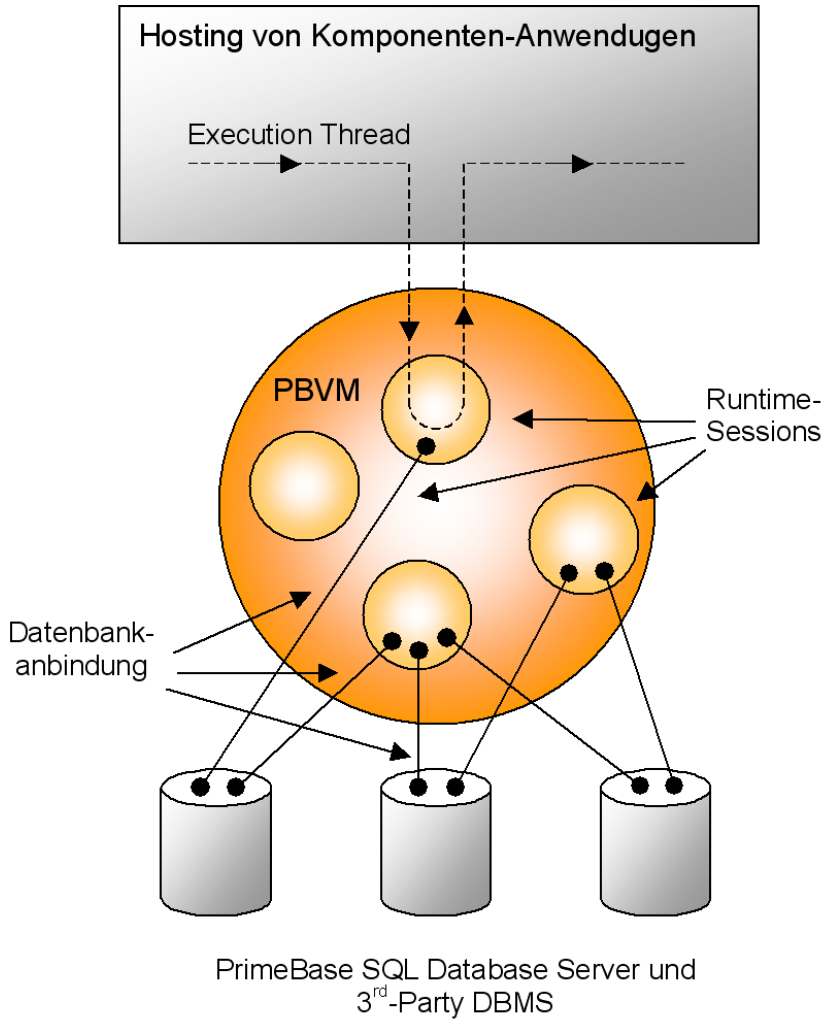

Figure 4: Internal design of the PBVM

The boot phase is divided into two phases itself. The first phase is the initialization phase, and the second phase is the startup phase. In the initialization phase the system "clone session" is duplicated to create a new session. In the startup phase the new session loads the code specified in the 'startup.sys' system control file (more details on the system files are given in the section "The PrimeBase Environment").

The system clone session is created before all other sessions are created by the virtual machine. This session is created by loading the code specified in the 'initialize.sys' system control file.

Both 'startup.sys' and 'initialize.sys' may load further code and data as required by the program to be run by the virtual machine.

Since all sessions are duplicates of the system clone session, all sessions initially

contain the code and data specified in the 'initialize.sys' system control file. Since 'initialize.sys' will generally load all the code required by the application, the cloning mechanism allows the virtual machine to create a new program instance almost instantly.

This is a basic requirement of Web-based applications. When the PBVM is used to run an Internet application, a new program instance is created for each user that connects to the system. At this moment the virtual machine creates and boots a new session. This is all done fast enough, so that the user does not notice any significant delay before the first requested page is returned to her/his browser.

When a session is terminated (closed), the code specified in the system 'shutdown.sys' system control file is loaded and executed. This allows the program instance to save state information in persistent storage before termination.

Connections

Also illustrated in Figure 4 are the connections from the runtime sessions to the various DBMSs (Database Management Systems) connected to the PrimeBase System. This includes the PrimeBase SQL Database Server and other 3rd party DBMSs that can be accessed via the PrimeBase Open Server Interface.

Note that each session may have zero, one or more connections to various DBMSs. Each connection is known as an "open DBMS".

Just how many connections each session requires depends on the program executed by the virtual machine. It also depends on the API used by the hosting application. For example, APIs such as ODBC and JDBC have no session concept. These APIs only deal with connections. As a result, the ODBC call used to create a connection is translated into a call to the virtual machine that creates a session with one connection. In this way, the session concept of the PBVM is transparent to the ODBC user.

As mentioned above, a session may also be created without any connections. The program instance running in the session will function normally, unless it requires access to database information. At any stage, a session may open a connection or additional connections using the OPEN DBMS statement provided by PBT.

A connection definition specifies all the information a session needs to open a new connection. This includes information such as which communications protocol (TCP/IP, AppleTalk, Shared Memory) to use, the host name of the server machine, the server name or port number, etc. These are called connection options. Connection definitions are stored in the 'connect.def' system control file. Each connection definition has a unique name called the connection alias.

The connection alias may be used when opening a connection, either in the OPEN DBMS statement or in the API call to create a new session. The alternative would be to explicitly specify all the connection options required for the connection.

4. The PrimeBase Environment

When you install a component of the PrimeBase system on your machine it is installed together with an "environment" in which the component will run. The environment consists of several folders (directories) containing system and control files. PrimeBase components are "pre-installed", because all that is required for installation, is to copy the executables to the host machine together with the folders and files that compose the environment. The PrimeBase component may then be started.

In this section we discuss the aspects of the PrimeBase environment that are common to all PrimeBase components. The concepts described here will help you to deploy and configure a system that consists of any number of PrimeBase components, and help you to quickly understand an existing setup.

Local and Global Locations

For every PrimeBase environment, two locations are of importance. The one is called the local location, and the other is the global location. The part of the environment in the local location is also known as the "Local Environment" and the part

of the environment in the global location is known as the "Global Environment".

When a PrimeBase component searches for a control file or folder, it first checks the local location and then the global location for the file. This means that controls files, placed in the local location, override files placed in the global location.

The actual position of these locations within the file system depends on the host platform.

Local locations:

- On *UNIX* and *Windows* platforms, the local location is always the "current working directory".
- On the *Macintosh* the local location is the location of the application.

Global locations:

- Under *UNIX* the default global location is /usr/local/primebase. The UNIX environment variable PRIMEBASE_HOME can be used to specify another location.
- Under *Windows* the global location is the system "Windows Directory".
- On the *Macintosh*, the system "Preferences Folder" is used.

As mentioned above, all PrimeBase components ship pre-installed. This means that all the control files needed to run the component are shipped pre-installed in the local location of the component. In this case, the local location, which is the current working directory under Windows and UNIX, is considered to be the same location as the application.

The PrimeBase System Folders

In each location you will find one or more PrimeBase system folders. The main system folders in the PrimeBase environment are illustrated in figure 5 on page 12. The PrimeBase system "Setup Folder" contains the system and control files for all PrimeBase components. In addition to this, the PrimeBase SQL Database has a system folder called the "DataServer Root Location" and the PrimeBase Application Server has a system folder called the "HTML Page Location".

The PrimeBase system Setup Folder is always called 'setup' when located in the local location. This folder is also known as the Local Setup Folder. The name of the Setup Folder, when located in the global location (the Global Setup Folder), depends on the host platform:

- Under *UNIX* the system Setup Folder is called 'primebase.setup'.
- Under *Windows* the system Setup Folder is called 'PB-SETUP'.
- On the *Macintosh*, the system Setup Folder is called 'PrimeBase Setup'.

These distinctive names for the Global Setup Folder ensure that the folder may be easily located on every machine.

The "HTML Page Location" and the "DataServer Root Location" are always located in the local location, since the files stored in these locations cannot be shared between multiple PrimeBase components. The "HTML Page Location" folder is called 'docs' on all platforms.

The name of the "DataServer Root Location" folder depends on the platform:

- Under UNIX the "DataServer Root Location" folder is called 'db'.
- Under Windows the "DataServer Root Location" folder is called 'DATABASE'.
- On the Macintosh, the "DataServer Root Location" folder is called 'Databases'.

Note that it is possible that a number of PrimeBase components share the same local location, however the components must all be of a different type. For example, two instances of the PrimeBase SQL Database Server cannot share the same local location.

.

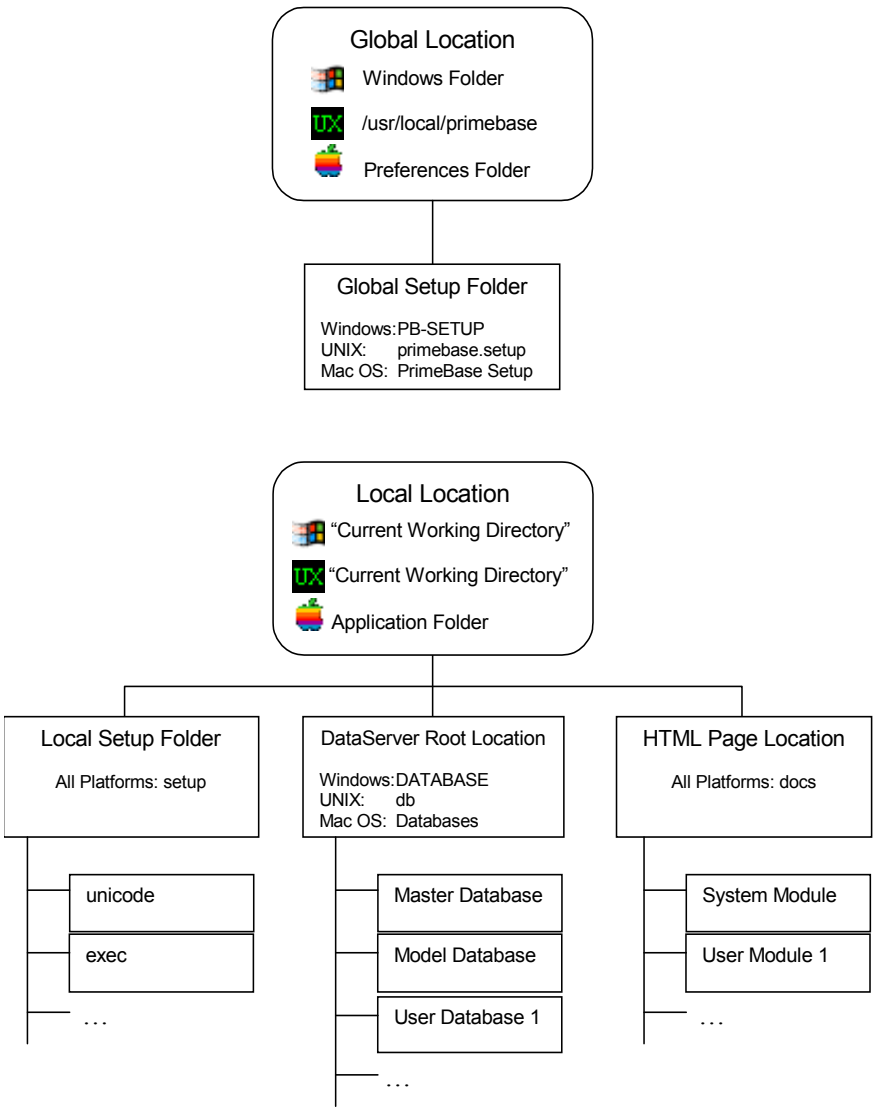

Figure 5: PrimeBase System Folders

The PrimeBase Setup Folder contains a number of sub-directories. The 'unicode' sub-directory contains the Unicode mapping files used by the system. These are standard mapping files available from http://www.unicode.org.

The 'exec' sub-directory is the default location for executable files used by the PrimeBase Virtual Machine. The directory contains the system boot code mentioned before, including 'initialize.sys', startup.sys' and 'shutdown.sys'.

The "DataServer Root Location" folder is the default location for databases mounted by the PrimeBase SQL Database Server. Each sub-directory in this folder is a database. Two system databases, the 'Master' and the 'Model' databases are always present.

The "HTML Page Location" folder contains the modules and documents served by the PrimeBase Application Server. Each sub-directory in this folder is a module. A module is installed by simply moving it into this folder. The System Module provides the administration interface for PBAS.

System Files & Parameters

The PrimeBase system files store parameters and other information that controls various aspects of the PrimeBase system. These files are located in the Prime-

Base system Setup Folder or sub-directories of this folder.

Each component of the PrimeBase system has a system "Environment File". The Environment File stores the values of various system parameters of the component. The name of Environment File corresponds to the name of the PrimeBase component, and ends with a '.env' extension. For example, the Environment File of the PrimeBase Virtual Machine is 'pbvm.env'.

The PrimeBase SQL Database Server stores a copy of its system parameters in the Master database. The values in the environment file, in this case, are only used for booting the server. After PBDS has booted, the values of the parameters in the Master database are used.

The values in the various Environment Files are stored in binary format. They can be edited with the PrimeBase Environment Editor. In the case of the PrimeBase SQL Database Server, the system parameters must be set directly in the Master database.

As an alternative to the Environment Editor, both the PrimeBase SQL Database Server and the PrimeBase Application Server provide a Web-based interface for setting system parameters (the PBDS Admin Server and the PBAS System Module).

As mentioned before, the "connection definitions" used by the system are stored in a system file called 'connect.def'. This is a text file that can be edited with a standard text editor. Each line of the file represents a connection definition. Connection definitions may also be created or modified using the PrimeBase Automation Client, the PBDS Admin Server or the PBAS System Module.

Connection definitions are used by the PrimeBase Virtual Machine or the PBDS Admin Server to connect to a PrimeBase SQL Database Server or some other DBMS via the PrimeBase Open Server Interface.

Further information on the system control files used by the PrimeBase components is given in the documentation for the various components.

Shared Memory & Component Names

A block of shared memory is used by the PrimeBase components on a particular machine to exchange system information. PrimeBase can also use the share memory area for standard communications in place of protocols such as TCP/IP.

Each PrimeBase component that runs as an application in the background has a unique name and "publishes" itself with this name in the shared memory area.

The various PrimeBase components have the following names by default:

- The PrimeBase SQL Database Server has the name "PrimeServer".
- The name of the PrimeBase Application Server is "PrimeBase".
- The PrimeBase Open Server has the name "OpenServer".
- The name of the PrimeBase Automation Client is "AutoTask1".

Since the component names must be unique, you need to change the default name if you wish to run multiple instances of a PrimeBase component on one machine. Note that the automation client only requires a unique name if it is not running in normal interactive mode.

The component name is a standard system parameter and can be set like other system parameters in the environment file, or in the Master database in the case of PBDS.

The PrimeBase Console

The PrimeBase console provides a text-based interface to the various PrimeBase components. Depending on the PrimeBase component the console allows you to perform administration tasks and retrieve status information about the component. In addition to this, components can be configured to print error, warnings and other messages to the console. This information can be used for debugging and trouble shooting.

When you start the PrimeBase console, you will be presented with a list of PrimeBase components running on the machine. You can then choose a component and the console will attach to that component.

Commands entered on the console begin with a # character. All components support #help, which prints a list of available commands. You can shutdown a component using the console by entering #halt.

In addition to the # commands, the console of the PrimeBase SQL Database Server allows you to login and enter and run PBT programs interactively. This allows you to perform sophisticated tasks, such as backup or database re-organization directly from the PBDS console.

You can enter CTRL-R to quit the console at any time, leaving the application running in the background.

Configurations

The PrimeBase components can be shipped and deployed in a number of different configurations. The provides you with a great deal of flexibility in planning your system. Depending on your system requirements, the various components can be deployed as a shared library or as a stand-alone application.

As mentioned before, the PrimeBase SQL Database Server can be combined with the PrimeBase Virtual Machine as a shared library and linked directly to an application. This is known as the PBDS "runtime" version.

The runtime version of PBDS is used to deploy a so-called "compact system", which does not require a stand-alone database server. For example, the PrimeBase Application Server can be shipped with the runtime version of the PBDS. To access data in the database, over the Web, all you need to do is start the application server. Using the runtime, the application server has direct access to the data in the database.

The scenario is also possible, because the PrimeBase Application Server supports direct HTTP access. This means access to the data from the Web is possible without starting a Web server.

One disadvantage of the PBDS runtime version, however, is that browser-based administration of the database server is not possible in this configuration.

Also mentioned previously is the fact that the PrimeBase Open Server is available as a stand-alone application, or as part of the PrimeBase Virtual Machine.

All PrimeBase components are also designed to run on read-only media such as CD ROM. This means it is possible to ship a PrimeBase-based system that can be used without installation, by starting it directly on the CD.

5. Connectivity

PrimeBase offers a variety of tools to connect to the PrimeBase Database Server:

PrimeBase PHP Module

The PrimeBase PHP Module allows you to access PrimeBase SQL Database Server from the popular PHP system.

It supports all features of the PrimeBase SQL Database Server, including transactions and BLOBs (binary large objects). Also Prepared Statements are supported.

To help developers already familiar with PHP database programming the module implements a superset of the MySQL specific PHP functions.

The PrimeBase PHP Module is available for Mac OS X, Linux, Solaris, AIX and Windows.

PrimeBase Perl DBD Module

The PrimeBase Perl DBD Module allows you to access PrimeBase SQL Database Server from Perl.

The module supports all features of the PrimeBase SQL Database Server, includ-

ing transactions and BLOBs (binary large objects). Also Prepared Statements are supported.

The PrimeBase Perl DBD Module is available for MacOS, Mac OS X, Linux, Solaris, AIX and Windows.

PrimeBase REALbasic Plug-in

The PrimeBase REALbasic Plug-in allows you to access PrimeBase SQL Database Server from REALbasic. The plug-in supports all features of the PrimeBase SQL Database Server, including transactions and BLOBs (binary large objects).

The plug-in can be used to build applications for MacOS (PPC), MacOS (PPC, Carbon), Mac OS X (Carbon), and Windows

Using the PrimeBase REALbasic Plug-in the typical Client/Server configuration can be used for your REALbasic application. This means that there is a database server which gets contacted by client software to perform database accesses and returning the results to the client software. (An example of such a request is a client software requesting all customer entries for customers who live in a specific country from the database server.)

In addition to Client/Server configurations, REALbasic applications using the PrimeBase REALbasic plug-in can also be "switched over" to use the runtime version of the PrimeBase SQL Database Server.

The runtime version of the database server allows to embed the complete PrimeBase SQL Database Server system into applications. The Runtime Version will be automatically started along with the application as part of the process of the application. There will be no database server process started on the computer, thus the user will not notice the embedded database system, which nonetheless can be used inside the application as if it were a standalone database server.

The runtime version also allows to ship your application on read-only media like CDROMs or DVDs while still being able to start the application with a simple doubleclick - directly from CDROM or DVD.

Configuring the application to use the runtime version of the PrimeBase SQL Database Server is in the simplest case achieved by changing an external configuration file. No code or only few changes to the application source code itself are required.

PrimeBase Xtra for Macromedia Director

The PrimeBase Xtra for Macromedia Director provides Macromedia Director movies access to the PrimeBase SQL Database Server.

The PrimeBase Xtra is a "scripting Xtra", which means that it adds functionality to Lingo, the script language used by Director. It allows you to use the full set of features of the PrimeBase SQL Database Server in your Director movie, for example database transactions for ensuring data consistency, powerful and fast full-text search and complex data accesses available in the relational database model.

PrimeBase Xtra for Macromedia Director is supported on Mac OS and Windows.

Using the PrimeBase Xtra the typical Client/Server configuration can be used for your Director movie. This means that there is a database server which gets contacted by client software (in this case for example your movie) to perform database accesses and returning the results to the client software. (An example of such a request is a client software requesting all customer entries for customers who live in a specific country.)

In addition to Client/Server configurations, movies written with the PrimeBase Xtra can also be "switched over" to use the runtime version of the PrimeBase SQL Database Server. The runtime version of the database server gives you the ability to ship your Director movie on read-only media like CDROMs or DVDs while still being able to start the movie with a simple doubleclick - directly from CDROM or DVD, without requiring any installation steps.

Configuring the movie to use the runtime version of the PrimeBase SQL Database

Server is simply achieved by changing an external configuration file. No or only few source code changes to the Director movie itself are required.

6. Conclusion

PrimeBase offers a complete solution for the development of Web-based applications and system. By supporting numerous industry standards, the various components can be used on their own or easily integrated into an existing system. Additionally, PrimeBase supports a large number of hardware platforms and operating systems including Windows, LINUX, Sun Solaris, IBM AIX and Mac OS.

PrimeBase provides a solid foundation, with high performance and functionality combined with maximum ease of use. This makes PrimeBase ideal for building e-Business and other complex Internet solutions today.